Inspiring Excellence

Course Title: Programming Language II
Course Code: CSE 111
Lab Assignment no: 3 & 4 Merged

Task 1

Implement the design of the **Patient** class so that the following output is produced:

[For BMI, the formula is BMI = weight/height^2, where weight is in kg and height in meters]

Driver Code	Output
# Write your code here p1 = Patient("A", 55, 63.0, 158.0) p1.printDetails() print("====================") p2 = Patient("B", 53, 61.0, 149.0) p2.printDetails()	Name: A Age: 55 Weight: 63.0 kg Height: 158.0 cm BMI: 25.236340330075304 ==================== Name: B Age: 53 Weight: 61.0 kg Height: 149.0 cm BMI: 27.476239809017613

Task 2

Design a class Shape for the given code below.

• Write a class Shape.

• Write the required constructor that takes 3 parameters and initialize the instance variables accordingly.

• Write a method area() that prints the area.

Hint: the area method can calxculate only for the shapes: Triangle, Rectangle, Rhombus, and Square. So, you have to use conditions inside this method

For this task, assume that --

- for a triangle, the arguments passed are the base and height
- for a rhombus, the arguments passed are the diagonals
- for a square or rectangle, the arguments passed are the sides.

Driver Code	Output
# Write your code here triangle = Shape("Triangle",10,25) triangle.area() print("==========") square = Shape("Square",10,10) square.area() print("==========") rhombus = Shape("Rhombus",18,25) rhombus.area() print("==========") rectangle = Shape("Rectangle",15,30) rectangle.area() print("==========") trapezium = Shape("Trapezium",15,30) trapezium.area()	Area: 125.0 ========== Area: 100 ========== Area: 225.0 ========== Area: 450 ========== Area: Shape unknown

Task 3

Implement the design of the **Calculator** class so that the following output is produced:

Driver Code	Output
# Write your code here c1 = Calculator() print("==========") val = c1.calculate(10, 20, '+') print("Returned value:", val) c1.showCalculation() print("==========") val = c1.calculate(val, 10, '-') print("Returned value:", val) c1.showCalculation() print("==========") val = c1.calculate(val, 5, '*') print("Returned value:", val) c1.showCalculation() print("==========") val = c1.calculate(val, 16, '/') print("Returned value:", val) c1.showCalculation()	Calculator is ready! ========== Returned value: 30 10 + 20 = 30 ========== Returned value: 20 30 - 10 = 20 ========== Returned value: 100 20 * 5 = 100 ========== Returned value: 6.25 100 / 16 = 6.25

Task 4

Design the **Programmer** class in such a way so that the following code provides the expected output.

Hint:

- Write the constructor with appropriate printing and multiple arguments.
- Write the addExp() method with appropriate printing and argument.
- Write the printDetails() method

[You are not allowed to change the code below]

# Write your code here.	***OUTPUT:***
p1 = Programmer("Ethen Hunt", "Java", 10) p1.printDetails() print('--------------------------') p2 = Programmer("James Bond", "C++", 7) p2.printDetails() print('--------------------------') p3 = Programmer("Jon Snow", "Python", 4) p3.printDetails() p3.addExp(5) p3.printDetails()	Horray! A new programmer is born Name: Ethen Hunt Language: Java Experience: 10 years. -------------------------- Horray! A new programmer is born Name: James Bond Language: C++ Experience: 7 years. -------------------------- Horray! A new programmer is born Name: Jon Snow Language: Python Experience: 4 years. Updating experience of Jon Snow Name: Jon Snow Language: Python Experience: 9 years.

Task 5

Implement the design of the **UberEats** class so that the following output is produced:

[For simplicity, you can assume that a customer will always order exact 2 items]

Driver Code	Output
# Write your code here order1 = UberEats("Shakib", "01719658xxx", "Mohakhali") print("==========") order1.add_items("Burger", "Coca Cola", 220, 50) print("==========") print(order1.print_order_detail()) print("==========") order2 = UberEats ("Siam", "01719659xxx", "Uttara") print("==========") order2.add_items("Pineapple", "Dairy Milk", 80, 70) print("==========") print(order2.print_order_detail())	Shakib, welcome to UberEats! ========== ========== User details: Name: Shakib, Phone: 01719658xxx, Address: Mohakhali Orders: {'Burger': 220, 'Coca Cola': 50} Total Paid Amount: 270 ========== Siam, welcome to UberEats! ========== ========== User details: Name: Siam, Phone: 01719659xxx, Address: Uttara Orders: {'Pineapple': 80, 'Dairy Milk': 70} Total Paid Amount: 150

Task 6

Write a class called **Customer** with the required constructor and methods to get the following output.

Subtasks:

1. Create a class called Customer.
2. Create the required constructor.
3. Create a method called **greet** that works if no arguments are passed or if one argument is passed. *(Hint: You may need to use the keyword NONE)*
4. Create a method called **purchase** that can take as many arguments as the user wants to give.

[You are not allowed to change the code below]

# Write your codes for subtasks 1-4 here.	***OUTPUT:***
customer_1 = Customer("Sam") customer_1.greet() customer_1.purchase("chips", "chocolate", "orange juice") print("----------") customer_2 = Customer("David") customer_2.greet("David") customer_2.purchase("orange juice")	Hello! Sam, you purchased 3 item(s): chips chocolate orange juice ---------- Hello David! David, you purchased 1 item(s): orange juice

Task 7

Analyze the given code below to write **Cat** class to get the output as shown.
Hints:

- *Remember, the constructor is a special method. Here, you have to deal with constructor overloading which is similar to method overloading.*
- *You may need to use the keyword None*
- *Your class should have 2 variables*

[You are not allowed to change the code below]

#Write your code here	***OUTPUT***
c1 = Cat() c2 = Cat("Black") c3 = Cat("Brown", "jumping") c4 = Cat("Red", "purring") c1.printCat() c2.printCat() c3.printCat() c4.printCat() c1.changeColor("Blue") c3.changeColor("Purple") c1.printCat() c3.printCat()	White cat is sitting Black cat is sitting Brown cat is jumping Red cat is purring Blue cat is sitting Purple cat is jumping

Task 8

Design the **Student** class such a way so that the following code provides the expected output.

Hint:

- Write the constructor with appropriate default value for arguments.
- Write the dailyEffort() method with appropriate arguments.
- Write the printDetails() method. For printing suggestions check the following instructions.

    If hour <= 2 print 'Suggestion: Should give more effort!'
    If hour <= 4 print 'Suggestion: Keep up the good work!'
    Else print 'Suggestion: Excellent! Now motivate others.'

[You are not allowed to change the code below]

# Write your code here.	***OUTPUT:***
harry = Student('Harry Potter', 123) harry.dailyEffort(3) harry.printDetails() print('========================') john = Student("John Wick", 456, "BBA") john.dailyEffort(2) john.printDetails() print('========================') naruto = Student("Naruto Uzumaki", 777, "Ninja") naruto.dailyEffort(6) naruto.printDetails()	Name: Harry Potter ID: 123 Department: CSE Daily Effort: 3 hour(s) Suggestion: Keep up the good work! ========== Name: John Wick ID: 456 Department: BBA Daily Effort: 2 hour(s) Suggestion: Should give more effort! ========== Name: Naruto Uzumaki ID: 777 Department: Ninja Daily Effort: 6 hour(s) Suggestion: Excellent! Now motivate others.

Task 9

Implement the design of the **Batsman** class so that the following output is produced:

Hint: Batting strike rate (s/r) = runsScored / ballsFaced x 100.

Driver Code	Output
# Write your code here b1 = Batsman(6101, 7380) b1.printCareerStatistics() print("============================") b2 = Batsman("Liton Das", 678, 773) b2.printCareerStatistics() print("----------------------------") print(b2.battingStrikeRate()) print("============================") b1.setName("Shakib Al Hasan") b1.printCareerStatistics() print("----------------------------") print(b1.battingStrikeRate())	Name: New Batsman Runs Scored: 6101 , Balls Faced: 7380 ============================ Name: Liton Das Runs Scored: 678 , Balls Faced: 773 ---------- 87.71021992238033 ============================ Name: Shakib Al Hasan Runs Scored: 6101 , Balls Faced: 7380 ---------- 82.66937669376694

Task 10

Implement the design of the **Author** class so that the following output is produced:

Driver Code	Output
# Write your code here auth1 = Author('Humayun Ahmed') auth1.addBooks('Deyal', 'Megher Opor Bari') auth1.printDetails() print("==========") auth2 = Author() print(auth2.name) auth2.changeName('Mario Puzo') auth2.addBooks('The Godfather', 'Omerta', 'The Sicilian') print("==========") auth2.printDetails() print("==========") auth3 = Author('Paolo Coelho', 'The Alchemist', 'The Fifth Mountain') auth3.printDetails()	Author Name: Humayun Ahmed -------- List of Books: Deyal Megher Opor Bari ========== Default ========== Author Name: Mario Puzo -------- List of Books: The Godfather Omerta The Sicilian ========== Author Name: Paolo Coelho -------- List of Books: The Alchemist The Fifth Mountain

Task 11

Using **TaxiLagbe** apps, users can share a single taxi with multiple people.

Implement the design of the **TaxiLagbe** class so that the following output is produced:

Hint:

1. Each taxi can carry maximum 4 passengers

2. addPassenger() method takes the last name of the passenger and ticket fare for that person in an underscore (-) separated string.

Driver Code	Output
# Write your code here # Do not change the following lines of code. taxi1 = TaxiLagbe('1010-01', 'Dhaka') print('------------------------------') taxi1.addPassenger('Walker_100', 'Wood_200') taxi1.addPassenger('Matt_100') taxi1.addPassenger('Wilson_105') print('------------------------------') taxi1.printDetails() print('------------------------------') taxi1.addPassenger('Karen_200') print('------------------------------') taxi1.printDetails() print('------------------------------') taxi2 = TaxiLagbe('1010-02', 'Khulna') taxi2.addPassenger('Ronald_115') taxi2.addPassenger('Parker_215') print('------------------------------') taxi2.printDetails()	---------- Dear Walker! Welcome to TaxiLagbe. Dear Wood! Welcome to TaxiLagbe. Dear Matt! Welcome to TaxiLagbe. Dear Wilson! Welcome to TaxiLagbe. ---------- Trip info for Taxi number: 1010-01 This taxi can cover only Dhaka area. Total passengers: 4 Passenger lists: Walker, Wood, Matt, Wilson Total collected fare: 505 Taka ---------- Taxi Full! No more passengers can be added. ---------- Trip info for Taxi number: 1010-01 This taxi can cover only Dhaka area. Total passengers: 4 Passenger lists: Walker, Wood, Matt, Wilson Total collected fare: 505 Taka ---------- Dear Ronald! Welcome to TaxiLagbe. Dear Parker! Welcome to TaxiLagbe. ---------- Trip info for Taxi number: 1010-02 This taxi can cover only Khulna area. Total passengers: 2 Passenger lists: Ronald, Parker Total collected fare: 330 Taka

Task 12

Implement the design of the **Account** class so that the following output is produced:

Driver Code	Output
# Write your code here a1 = Account() print(a1.details()) print("----------") a1.name = "Oliver" a1.balance = 10000.0 print(a1.details()) print("----------") a2 = Account("Liam") print(a2.details()) print("----------") a3 = Account("Noah",400) print(a3.details()) print("----------") a1.withdraw(6930) print("----------") a2.withdraw(600) print("----------") a1.withdraw(6929)	Default Account 0.0 ---------- Oliver 10000.0 ---------- Liam 0.0 ---------- Noah 400.0 ---------- Sorry, Withdraw unsuccessful! The account balance after deducting withdraw amount is equal to or less than minimum. ---------- Sorry, Withdraw unsuccessful! The account balance after deducting withdraw amount is equal to or less than minimum. ---------- Withdraw successful! New balance is: 3071.0

Task 13

Implement the design of the **StudentDatabase** class so that the following output is produced:

GPA = Sum of (Grade Points * Credits)/ Credits attempted

Driver Code	Output
# Write your code here # Do not change the following lines of code. s1 = StudentDatabase('Pietro', '10101222') s1.calculateGPA(['CSE230: 4.0', 'CSE220: 4.0', 'MAT110: 4.0'], 'Summer2020') s1.calculateGPA(['CSE250: 3.7', 'CSE330: 4.0'], 'Summer2021') print(f'Grades for {s1.name}\n{s1.grades}') print('----------') s1.printDetails() s2 = StudentDatabase('Wanda', '10103332') s2.calculateGPA(['CSE111: 3.7', 'CSE260: 3.7', 'ENG101: 4.0'], 'Summer2022') print('----------') print(f'Grades for {s2.name}\n{s2.grades}') print('----------') s2.printDetails()	Grades for Pietro {'Summer2020': {('CSE230', 'CSE220', 'MAT110'): 4.0}, 'Summer2021': {('CSE250', 'CSE330'): 3.85}} ---------- Name: Pietro ID: 10101222 Courses taken in Summer2020: CSE230 CSE220 MAT110 GPA: 4.0 Courses taken in Summer2021: CSE250 CSE330 GPA: 3.85 ---------- Grades for Wanda {'Summer2022': {('CSE111', 'CSE260', 'ENG101'): 3.8}} ---------- Name: Wanda ID: 10103332 Courses taken in Summer2022: CSE111 CSE260 ENG101 GPA: 3.8

Task 14

```
1  class Test3:
2      def __init__(self):
3          self.sum, self.y = 0, 0
4      def methodA(self):
5          x, y = 2, 3
6          msg = [0]
7          msg[0] = 3
8          y = self.y + msg[0]
9          self.methodB(msg, msg[0])
10         x = self.y + msg[0]
11         self.sum = x + y + msg[0]
12         print(x, y, self.sum)
13     def methodB(self, mg2, mg1):
14         x = 0
15         self.y = self.y + mg2[0]
16         x = x + 33 + mg1
17         self.sum = self.sum + x + self.y
18         mg2[0] = self.y + mg1
19         mg1 = mg1 + x + 2
20         print(x, self.y, self.sum)
```

Write the output of the following code: t3 = Test3() t3.methodA() t3.methodA() t3.methodA() t3.methodA()	x	y	sum

Task 15

```
1  class Test5:
2      def __init__(self):
3          self.sum, self.y = 0, 0
4      def methodA(self):
5          x = 0
6          z = 0
7          while (z < 5):
8              self.y = self.y + self.sum
9              x = self.y + 1
10             print(x, self.y, self.sum)
11             self.sum = self.sum + self.methodB(x, self.y)
12             z += 1
13     def methodB(self, m, n):
14         x = 0
15         sum = 0
16         self.y = self.y + m
17         x = n - 4
18         sum = sum + self.y
19         print(x, self.y, sum)
20         return self.sum
```

Write the output of the following code:

```
t5 = Test5()
t5.methodA()
```

x	y	sum

Task 16

```
1  class FinalT6A:
2      def __init__(self, x, p):
3          self.temp, self.sum, self.y = 4, 0, 1
4          self.temp += 1
5          self.y = self.temp - p
6          self.sum = self.temp + x
7          print(x, self.y, self.sum)
8      def methodA(self):
9          x = 0
10         y = 0
11         y = y + self.y
12         x = self.y + 2 + self.temp
13         self.sum = x + y + self.methodB(self.temp, y)
14         print(x, y, self.sum)
15     def methodB(self, temp, n):
16         x = 0
17         temp += 1
18         self.y = self.y + temp
19         x = x + 3 + n
20         self.sum = self.sum + x + self.y
21         print(x, self.y, self.sum)
22         return self.sum
```

What is the output of the following code sequence?

```
q1 = FinalT6A(2,1)
q1.methodA()
q1.methodA()
```

x	y	sum

Task 17

```
1  class Test5:
2    def __init__(self):
3        self.sum = 0
4        self.y = 0
5    def methodA(self):
6        x=y=k=0
7        msg = [5]
8        while (k < 2):
9            y += msg[0]
10           x = y + self.methodB(msg, k)
11           self.sum = x + y + msg[0]
12           print(x ," " , y, " " , self.sum)
13           k+=1
14   def methodB(self, mg2, mg1):
15         x = 0
16         self.y += mg2[0]
17         x = x + 3 + mg1
18         self.sum += x + self.y
19         mg2[0] = self.y + mg1
20         mg1 += x + 2
21         print(x , " " ,self.y, " " , self.sum)
22         return mg1
```

What is the output of the following code sequence?

```
t1 = Test5()
t1.methodA()
t1.methodA()
t1.methodA()
```

x	y	sum

Task 18

```
1  class Test4:
2      def __init__(self):
3          self.sum, self.y = 0, 0
4      def methodA(self):
5          x, y = 0, 0
6          msg = [0]
7          msg[0] = 5
8          y = y + self.methodB(msg[0])
9          x = y + self.methodB(msg, msg[0])
10         self.sum = x + y + msg[0]
11         print(x, y, self.sum)
12     def methodB(self, *args):
13         if len(args) == 1:
14             mg1 = args[0]
15             x, y = 0, 0
16             y = y + mg1
17             x = x + 33 + mg1
18             self.sum = self.sum + x + y
19             self.y = mg1 + x + 2
20             print(x, y, self.sum)
21             return y
22         else:
23             mg2, mg1 = args
24             x = 0
25             self.y = self.y + mg2[0]
26             x = x + 33 + mg1
27             self.sum = self.sum + x + self.y
28             mg2[0] = self.y + mg1
29             mg1 = mg1 + x + 2
30             print(x, self.y, self.sum)
31             return self.sum
```

```
t3 = Test4()
t3.methodA()
t3.methodA()
t3.methodA()
t3.methodA()
```

x	y	sum

Task 19

```
1  class msgClass:
2      def __init__(self):
3          self.content = 0
4  class Q5:
5      def __init__(self):
6          self.sum = 1
7          self.x = 2
8          self.y = 3
9      def methodA(self):
10         x, y = 1, 1
11         msg = []
12         myMsg = msgClass()
13         myMsg.content = self.x
14         msg.append(myMsg)
15         msg[0].content = self.y + myMsg.content
16         self.y = self.y + self.methodB(msg[0])
17         y = self.methodB(msg[0]) + self.y
18         x = y + self.methodB(msg[0], msg)
19         self.sum = x + y + msg[0].content
20         print(x," ", y," ", self.sum)
21     def methodB(self, mg1, mg2 = None):
22         if mg2 == None:
23             x, y = 5, 6
24             y = self.sum + mg1.content
25             self.y = y + mg1.content
26             x = self.x + 7 +mg1.content
27             self.sum = self.sum + x + y
28             self.x = mg1.content + x +8
29             print(x, " ", y," ", self.sum)
30             return y
```

```
31          else:
32              x = 1
33              self.y += mg2[0].content
34              mg2[0].content = self.y + mg1.content
35              x = x + 4 + mg1.content
36              self.sum += x + self.y
37              mg1.content = self.sum - mg2[0].content
38              print(self.x, " ",self.y," ", self.sum)
39              return self.sum
```

What is the output of the following code sequence?

```
q = Q5()
q.methodA()
```

x	y	sum

Practice Task (20 - 25) Ungraded

Task 20

Design a **Student** class so that the following output is produced upon executing the following code

Driver Code	Output
# Write your code here **# Do not change the following lines of code.** s1 = Student() print("========================") s2 = Student("Carol") print("========================") s3 = Student("Jon", "EEE") print("========================") s1.update_name("Bob") s1.update_department("CSE") s2.update_department("BBA") s1.enroll("CSE110", "MAT110", "ENG091") s2.enroll("BUS101") s3.enroll("MAT110", "PHY111") print("##############################") s1.printDetail() print("========================") s2.printDetail() print("========================") s3.printDetail()	Student name and department need to be set ========== Department for Carol needs to be set ========== Jon is from EEE department ========== ########## Name: Bob Department: CSE Bob enrolled in 3 course(s): CSE110, MAT110, ENG091 ========== Name: Carol Department: BBA Carol enrolled in 1 course(s): BUS101 ========== Name: Jon Department: EEE Jon enrolled in 2 course(s): MAT110, PHY111

Task 21

Design a **Student** class so that the following output is produced upon executing the following code:
[Hint: Each course has 3.0 credit hours. You must take at least 9.0 and at most 12.0 credit hours]

Driver Code	Output
# Write your code here **# Do not change the following lines of code.** s1 = Student(“Alice”,“20103012”,“CSE”) s2 = Student(“Bob”, “18301254”,“EEE”) s3 = Student(“Carol”, “17101238”,“CSE”) print(“############################”) print(s1.details()) print(“############################”) print(s2.details()) print(“############################”) s1.advise(“CSE110”, “MAT110”, “PHY111”) print(“############################”) s2.advise(“BUS101”, “MAT120”) print(“############################”) s3.advise(“MAT110”, “PHY111”, “ENG102”, “CSE111”, “CSE230”)	############################ Name: Alice ID: 20103012 Department: CSE ############################ Name: Bob ID: 18301254 Department: EEE ############################ Alice, you have taken 9.0 credits. List of courses: CSE110, MAT110, PHY111 Status: Ok ############################ Bob, you have taken 6.0 credits. List of courses: BUS101, MAT120 Status: You have to take at least 1 more course. ############################ Carol, you have taken 15.0 credits. List of courses: MAT110, PHY111, ENG102, CSE111, CSE230 Status: You have to drop at least 1 course.

Task 22

Write the **Hotel** class with the required methods to give the following output as shown.

Driver Code	Output
# Write your code here **# Do not change the following lines of code.** h = Hotel("Lakeshore") h.addStuff("Adam", 26) print("==========") print(h.getStuffById(1)) print("==========") h.addGuest(“Carol”,35,”123”) print("==========") print(h.getGuestById(1)) print("==========") h.addGuest("Diana", 32, “431”) print("==========") print(h.getGuestById(2)) print("==========") h.allStaffs() print("==========") h.allGuest()	Staff With ID 1 is added ========== Staff ID: 1 Name: Adam Age: 26 Phone no.: 000 ========== Guest With ID 1 is created ========== Guest ID: 1 Name: Carol Age: 35 Phone no.: 123 ========== Guest With ID 2 is created ========== Guest ID: 2 Name: Dianal Age: 32 Phone no.: 431 ========== All Staffs: Number of Staff: 1 Staff ID: 1 Name: Adam Age: 26 Phone no: 000 ========== All Guest: Number of Guest: 2 Guest ID: 1 Name: Carol Age: 35 Phone no.: 123 Guest ID: 2 Name: Dianal Age: 32 Phone no.: 431

Task 23

Write the **Author** class with the required methods to give the following outputs as shown.

Driver Code	Output
# Write your code here **# Do not change the following lines of code.** a1 = Author() print("==========") a1.addBook("Ice", "Science Fiction") print("==========") a1.setName("Anna Kavan") a1.addBook("Ice", "Science Fiction") a1.printDetail() print("==========") a2 = Author("Humayun Ahmed") a2.addBook("Onnobhubon", "Science Fiction") a2.addBook("Megher Upor Bari", "Horror") print("==========") a2.printDetail() a2.addBook("Ireena", "Science Fiction") print("==========") a2.printDetail() print("==========")	========== A book can not be added without author name ========== Number of Book(s): 1 Author Name: Anna Kavan Science Fiction: Ice ========== ========== Number of Book(s): 2 Author Name: Humayun Ahmed Science Fiction: Onnobhubon Horror: Megher Upor Bari ========== Number of Book(s): 3 Author Name: Humayun Ahmed Science Fiction: Onnobhubon, Ireena Horror: Megher Upor Bari ==========

Task 24

Implement the design of the **Hospital, Doctor and Patient** class so that the following output is produced:

Driver Code	Output
# Write your code here **# Do not change the following lines of code.** h = Hospital("Evercare") d1 = Doctor("1d","Doctor", "Samar Kumar", "Neurologist") h.addDoctor(d1) print("==========") print(h.getDoctorByID("1d")) print("==========") p1 = Patient("1p","Patient", "Kashem Ahmed", 35, 12345) h.addPatient(p1) print("==========") print(h.getPatientByID("1p")) print("==========") p2 = Patient ("2p","Patient", "Tanina Haque", 26, 33456) h.addPatient(p2) print("==========") print(h.getPatientByID("2p")) print("==========") h.allDoctors() h.allPatients()	========== Doctor's ID: 1d Name: Samar Kumar Speciality: Neurologist ========== ========== Patient's ID: 1p Name: Kashem Ahmed Age: 35 Phone no.: 12345 ========== ========== Patient's ID: 2p Name: Tanina Haque Age: 26 Phone no.: 33456 ========== All Doctors: Number of Doctors: 1 {'1d': ['Samar Kumar', 'Neurologist']} All Patients: Number of Patients: 2 {'1p': ['Kashem Ahmed', 35, 12345], '2p': ['Tanina Haque', 26, 33456]}

Task 25

Design the **Vaccine** and **Person** class so that the following expected output is generated.
[N.B: Students will get vaccines on a priority basis. So, age for students doesn't matter**]**

Driver Code	Output
# Write your code here astra = Vaccine("AstraZeneca", "UK", 60) modr = Vaccine("Moderna", "UK", 30) sin = Vaccine("Sinopharm", "China", 30) p1 = Person("Bob", 21, "Student") print("==========") p1.pushVaccine(astra) print("==========") p1.showDetail() print("==========") p1.pushVaccine(sin, "2nd Dose") print("==========") p1.pushVaccine(astra, "2nd Dose") print("==========") p1.showDetail() print("==========") p2 = Person("Carol", 23, "Actor") print("==========") p2.pushVaccine(sin) print("==========") p3 = Person("David", 34) print("==========") p3.pushVaccine(modr) print("==========") p3.showDetail() print("==========") p3.pushVaccine(modr, "2nd Dose")	========== 1st dose done for Bob ========== Name: Bob Age: 21 Type: Student Vaccine name: AstraZeneca 1st dose: Given 2nd dose: Please come after 60 days ========== Sorry Bob, you can't take 2 different vaccines ========== 2nd dose done for Bob ========== Name: Bob Age: 21 Type: Student Vaccine name: AstraZeneca 1st dose: Given 2nd dose: Given ========== ========== Sorry Carol, Minimum age for taking vaccines is 25 years now. ========== ========== 1st dose done for David ========== Name: David Age: 34 Type: General Citizen Vaccine name: Moderna 1st dose: Given 2nd dose: Please come after 30 days ========== 2nd dose done for David